A case study of policy synthesis for swarm robotics

A case study of policy synthesis for swarm robotics

Paul Piho[1][0000−0002−4072−1000] and Jane Hillston[1][0000−0003−4914−9255]

University of Edinburgh, Edinburgh, UK

Abstract. Continuous time Markov chain models, derived from process algebraic descriptions of systems are a powerful method for studying the dynamics of collective adaptive systems. Here, we study a formal modelling framework, based on the CARMA process algebra, where information about the possible control actions of individual components in such systems can be incorporated in the process algebraic description. The formal semantics for such specifications are defined to give rise to continuous time Markov decision processes. Here we show how, together with a given specification of desired collective behaviour, such models can be readily treated as stochastic policy or control synthesis problems. This is demonstrated through an example scenario from swarm robotics.

1 Introduction

Computational modelling and simulation approaches provide a useful set of tools for studying complex dynamics of both man-made and natural collective systems. Various formal modelling approaches have been proposed to simplify the creation of such models. In particular stochastic process algebras with continuous time Markov chain (CTMC) semantics [18, 3, 14], have provided a powerful high-level framework for modelling collective systems, allowing compositional definitions of complex models and formal semantics for automation of model creation.

Process algebra-based models have been used to study a variety of phenomena in literature in order to better understand the processes involved or predict the real life performance of the system. Stochastic process algebras with underlying CTMC-based semantics lend themselves well to numerical or statistical analysis as well as various model-checking methods proposed over the years [15, 16]. In the context of man-made or engineered systems the interesting questions often relate to policy or parameter synthesis problems. In particular, how the components in such systems should be designed so that a system level objective is achieved. The link between high-level process algebraic models and the related policy synthesis models is usually not made explicit. In this paper we present a swarm robotics-inspired case study where this connection is made explicit by incorporating the information about control actions or possible choice of parameters into the process algebraic description of the system.

We consider an existing stochastic process algebra CARMA [18] which has previously successfully been applied to a range of application domains like pedestrian movement [12], urban transportation services [25], availability of cloud

services [20] and ambulance deployment [11]. The language features a set of communication primitives that, in conjunction with attribute-based filtering of communication partners, are capable of capturing a versatile set of communication behaviours. The set of communication primitives in CARMA correspond to broadcast and unicast, making it particularly suitable for open collectives where the participants of the communications cannot be known in advance.

The aim in this paper is to demonstrate through an example scenario from swarm robotics [7, 19] that processes algebraic constructions in CARMA lend themselves well to stochastic policy or parameter synthesis problems. Swarm robots provide rich modelling examples in the context of collective systems, since the directly controllable behaviours and interactions are those of individual robots but the design goals for the systems are phrased in terms of the aggregate behaviour of the entire collective rather that of individual robots. To that end, CARMA can be equipped with continuous time Markov decision processes (CTMDPs) based semantics [22] providing a natural formalisation of policy synthesis problems and bridging the gap between the formal high-level modelling and policy synthesis problems for collectives such as robot swarms. The contribution of this paper is to illustrate these constructions with a case study and show how ideas from formal modelling and policy synthesis come together in a framework for stochastic control or parameter synthesis problems.

The paper is structured as follows. In Section 2, as background, we introduce the notions of CTMDPs and population CTMDPs. In Section 3 we give a brief overview of the CARMA-based modelling framework for policy synthesis through a simple example. In Section 4 we present the swarm robotics-inspired case study. Finally we end the paper with related work and conclusions in Sections 5 and 6.

2 Background

The underlying mathematical model considered in this paper is a CTMDP. In particular, we consider a high-level formal modelling framework where the constructed models can be related to a CTMDP. To start let us give the definition of a CTMDP and introduce the related policy or parameter synthesis problems.

Definition 1. *A continuous-time Markov decision process (CTMDP) is defined by the tuple* $\{\mathcal{S}, \mathcal{A}, q(i, j \mid a)\}$ *where* $\mathcal{S}$ *is the countable set of states,* $\mathcal{A}$ *set of actions and* $q(i, j \mid a)$ *gives the transition rates* $i \to j$ *given the control action* $a \in \mathcal{A}$*. We use* $\mathcal{A}(i)$ *to denote the set of feasible actions in state* i*.*

The evolution of CTMDPs is described by the following: after the process reaches some state and an action is chosen, the process performs a transition to the next state depending only on the current state and the chosen action. The time it takes for state transitions to happen is governed by an exponential distribution with a rate given by the function q in Definition 1. The actions at every such step are chosen according to some policy as defined below.

Definition 2. *A policy is a measurable function* $\psi : \mathbb{R}_{\geq 0} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ *which for every time* $t \in \mathbb{R}_{\geq 0}$*, state* $i \in \mathcal{S}$ *and action* $a \in \mathcal{A}(i)$ *assigns a probability*

$\psi(t, i, a)$ that the action a is chosen in i at time t. In other words, policy ψ defines a distribution over actions in any state of the CTMDP at time t. We call a policy where, for every $t \in \mathbb{R}_{\geq 0}$ and $i \in S$, we have that $\psi(t, i, a) \in \{0, 1\}$, a deterministic policy. A policy ψ independent of t is a stationary policy.

Note that fixing a policy ψ resolves the non-determinism in the model and since the times that state transitions take are exponentially distributed the result is a continuous time Markov chain.

When we consider a system consisting of a large number of components with identical behaviours it is often convenient to consider a special case of CTMDPs.

Definition 3. *A population CTMDP (pCTMDP) is a tuple $(\mathbf{X}, \mathcal{T}, \mathcal{A}, \beta)$ where:*

- *$\mathbf{X} = (X_1, \cdots, X_n) \in \mathcal{S} = \mathbb{Z}^n_{\geq 0}$ where each X_i takes values in a finite domain $\mathcal{D}_i \subset \mathbb{Z}_{\geq 0}$.*
- *β is a function such that $\beta(a, \mathbf{X})$ returns a boolean value indicating whether action $a \in \mathcal{A}$ is available from state $\mathbf{X}$.*
- *$\mathcal{T}$ is a set of transitions of the form $\tau = (a, \mathbf{v}_\tau, r_\tau(\mathbf{X}))$ such that $\beta(a, \mathbf{X}) = 1$, $\mathbf{v}_\tau$ is an update vector specifying that the state after execution of transition τ is $\mathbf{X} + \mathbf{v}_\tau$ and $r_\tau(\mathbf{X})$ is a rate function.*

In order to give semantics to the above definition of a population CTMDP we associate it with the equivalent CTMDP in the following way:

- the state and action space of the corresponding CTMDP is the same as for the population CTMDP.
- the set of feasible actions for state $\mathbf{i} \in \mathcal{S}$, denoted $\mathcal{A}(\mathbf{i})$, is defined by

$$\mathcal{A}(\mathbf{i}) = \{a \in \mathcal{A} \mid \beta(a, \mathbf{i}) = 1\}.$$

- the rate function q is defined as

$$q(\mathbf{i}, \mathbf{j} \mid a) = \sum_{\tau \in \mathcal{T}, \tau = (a, \mathbf{v}_\tau, r_\tau(\mathbf{j})), \mathbf{i} = \mathbf{j} + \mathbf{v}_\tau} r_\tau(\mathbf{i}).$$

To form a policy synthesis problem for a given CTMDP we need a reward or a cost function which maps a chosen policy to a real value. A common approach for defining a reward function, for example, is as a function of the expected behaviour of the resulting CTMC.

3 Carma-C for policy synthesis

Carma [18] is a stochastic process algebra for quantitative modelling of collective adaptive systems. It supports the specification of complex stochastic behaviour, based on continuous time Markov chains, in a compositional way. In particular, each component in Carma consists of a process definition P and a local store γ. Carma models then consist of a collective N, composed of individual components or agents C, operating in an environment $\mathcal{E}$. This structure of Carma models is illustrated in Figure 1.

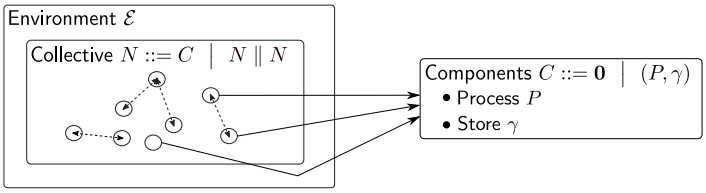


Fig. 1: Illustration of the structure of Carma models.

The Function Labelled Transition Systems (FuTS) style operational semantics [10] of Carma, as described in [18], give rise to a labelled transition system which can be simulated directly or translated into a CTMC for numerical analysis. The aim of this section is to introduce the extension Carma-C [22] for specifying policy synthesis problems. The underlying CTMDP model allows us to specify non-deterministic behaviour, corresponding to different possible action choices, in a Carma-C model. The approach is to incorporate the non-determinism into the store definitions. In Carma each attribute in store refers to a single value that is used in the semantics, for example, to evaluate guards or filter communication partners. Instead, in Carma-C we relax this construction and allow store attributes to refer to value domains which leads to non-determinism over a range of possible behaviours. Note that a parametric CTMC would be another reasonable choice for the underlying semantics that account for non-determinism. The semantics of the languages would not change much in that case as the construction of the parameter space of a CTMC from the high-level model description is analogous to the construction of the action space.

In the following we describe Carma-C and as a running example present the model for the case study in Section 4.1. This example model is outlined below.

Example 1. The model considers a simple robot swarm where an exploration phase is modelled by a random walk on the graph structure in Figure 2. The swarm attempts to discover and gather at the target location (x, y). This is modelled by an exploration followed by an aggregation phase. The switch between the two phases happens via broadcast communication. When any single robot detects the target at (x, y) it broadcasts this knowledge to the rest of the swarm.

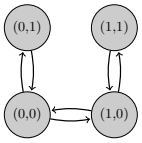


Fig. 2: Spatial structure for the example.

3.1 Local store

Let us start by considering the local stores of components. The local store is used to hold the attributes of an individual component. For example, a location

attribute of a component can be used to implement communication within a given range. CARMA and CARMA-C differ in their treatment of the local store — in CARMA-C we use the store attributes to also specify the action space of the underlying CTMDP. By associating each attribute in a store with a value domain rather than a single value, we introduce the non-determinism required to establish an action space. Here, the local store of each of the robots can be defined to hold attributes for location, denoted loc, set of known target locations, denoted target. The robustness parameter, succp, taking values in the interval $[0, 1]$, models how reliably the robots move when navigating towards a target.

$$\gamma_l = \{\mathsf{loc} \mapsto \{(0,0)\}, \mathsf{target} \mapsto \{\emptyset\}, \mathsf{succp} \mapsto [0,1]\}$$

The location loc and target target are defined to have singleton value domains consisting of pair $(0, 0)$ and the empty set respectively. The robustness parameter succp on the other hand takes values in the real interval $[0, 1]$.

In the semantics of CARMA-C the available control actions in the underlying CTMDP are associated with the possible ways we can refine the defined store to correspond to single values from the defined value domains. In particular, a control action f from a state of the CARMA-C system resolves all non-determinism in the descriptions. As an example, we can suppose that a given control action f applied to γ_l gives the following: $f(\gamma_l) = \{\mathsf{loc} \mapsto (0,0), \mathsf{target} \mapsto \emptyset, \mathsf{succp} \mapsto 0.7\}$. This control action results in a local store of a component at location $(0, 0)$ with no known targets. The parameter succp, when a target is known, captures the probability of the component successfully moving towards it.

3.2 Processes

Next let us consider the process definitions P. The processes are composed of action primitives corresponding to input and output actions for broadcast and unicast communication. Note that broadcast in CARMA is non-blocking — the output action is executed even if there is no component able to receive the message. Unicast on the other hand is blocking. Unicast output is denoted by $\alpha^* [\pi_s] \langle \vec{e} \rangle \sigma$ while unicast input is denoted by $\alpha [\pi_s] \langle \vec{e} \rangle \sigma$. Similarly broadcast output is denoted by $\alpha^* [\pi_s] \langle \vec{e} \rangle \sigma$ while broadcast input is denoted by $\alpha^* [\pi_r] (\vec{x}) \sigma$.

The following notation is used

- α is an action type which is used to distinguish between different actions.
- π_s, π_r, π denote boolean predicates that have to be satisfied before the action can be executed. As mentioned previously, the communication in CARMA and CARMA-C is attribute-based — guards are used to filter out communication partners based on attributes such as location or communication range.
- e is an expression built using appropriate combinations of values, attributes and variables. In the semantics, the expressions are evaluated over the sending component's local store and passed on to the receiving component.
- x is a variable which takes on the values that were communicated to the receiving process by the sender.

– σ is a function from $\Gamma \to Dist(\Gamma)$ where $Dist(\Gamma)$ is the set of distributions over the set of possible stores Γ. The function σ thus denotes a store *update* and defines how the given store is changed as a result of an action.

The processes are composed via the standard constructs — action prefix (.), choice (+), and parallel composition ($\|$). The behaviour of processes can be further modified by setting guards on processes. As an example, we can consider the following processes that we use to model the scenario in Example 1.

$$\begin{aligned} \mathit{Explore} \stackrel{\text{def}}{=} & [\pi_r]\mathit{random}^*[\circ]\langle\circ\rangle\{\mathsf{loc} \mapsto R(\mathsf{loc})\}.\mathit{Explore} \\ & + [\pi_d]\mathit{directed}^*[\circ]\langle\circ\rangle\{\mathsf{loc} \mapsto D(\mathsf{loc})\}.\mathit{Explore} \\ & + [\pi_r]\mathit{sense}^*[\circ]\langle\mathsf{loc}\rangle\{\mathsf{target} \mapsto \mathsf{target} \cup \{\mathsf{loc}\}\}.\mathit{Explore} \\ \mathit{Listen} \stackrel{\text{def}}{=} & [\pi_r]\mathit{sense}^*[\circ](\{(x,y)\})\{\mathsf{target} \mapsto \mathsf{target} \cup \{(x,y)\}\}.\mathit{Listen} \\ \mathit{Robot} \stackrel{\text{def}}{=} & \mathit{Explore} \parallel \mathit{Listen} \end{aligned}$$

Our example can then be modelled by the processes illustrated in Figure 3. The broadcast actions *random** and *directed** describe a random walk and a directed walk towards (x, y) respectively. Despite being defined as broadcast actions, neither of these actions have any effect on the other robots in the collective because the outgoing message is set to be empty. The guards π_r and π_d check whether the target location is known or not and make sure only one of the actions *random** and *directed** is enabled at a time. The guards are evaluated conditionally on a chosen control action f in the following way.

$$\pi_r = \begin{cases} \mathit{true} & \text{if } f(\gamma)(\mathsf{target}) = \emptyset. \\ \mathit{false} & \text{otherwise.} \end{cases} \qquad \pi_d = \begin{cases} \mathit{true} & \text{if } f(\gamma)(\mathsf{target}) = \{(x,y)\}. \\ \mathit{false} & \text{otherwise.} \end{cases}$$

The *sense** action models the detection of the target location. In particular, the broadcast output action models the robot detecting and sending the target location to the rest of the swarm. The corresponding broadcast input action models the robot's ability to receive such a message.

The actions *random** and *directed** change the loc attribute of the robot component according to functions R and D respectively. The (random) function R corresponds to the next location being selected uniformly from the set of available next locations defined by the graph structure in Figure 2. Similarly, D corresponds to the next location taking the robot closer to the target with some probability p, specified by the robustness attribute succp, and to one of

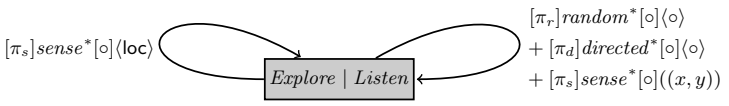


Fig. 3: Behaviour of individual *Robot* components.

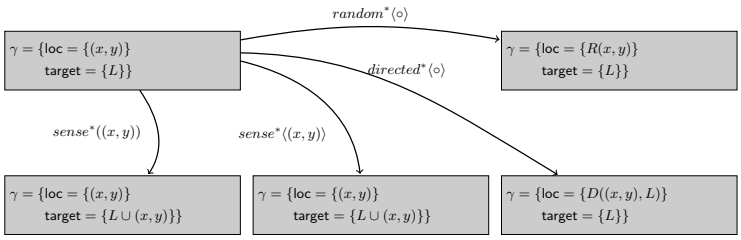


Fig. 4: Local component store changes induced by actions.

the other directly connected locations with probability $1 - p$. This defines a distribution over the possible unresolved local stores the components can evolve to and models unreliable navigation. The $sense^*$ action updates the set of target locations with the current location of the sending robot.

The functions R, D as well as the set operation $\cup$ are applied element-wise to all elements in the relevant value domains as illustrated in Figure 4. For example, consider the update $\mathsf{loc} \mapsto R(\mathsf{loc})$. This means that the function R is applied to every element in the value domain for loc. The store update for a given initial store are illustrated in Figure 4.

3.3 Environment

Finally we are going to address the environment. An environment is defined by a *global store* γ_g that models the overall state of the system and an *evolution rule* ρ. The global store is defined similarly to the local store. To continue the example, we define two global store variables — one corresponding to the rate at which the actions $random^*$ and $directed^*$ happen, denoted mover. We specify the value domain for this variable to be $[0, \infty)$. Similarly, we specify the value domain for the store attribute senser, corresponding to the rate of the action $sense^*$, to be $[0, \infty)$. Finally, we specify an attribute that keeps track of the location of the target. In particular,

$$\gamma_g = \{\mathsf{mover} \mapsto [0, \infty), \mathsf{senser} \mapsto [0, \infty), \mathsf{tloc} \mapsto \{(1, 1)\}\}.$$

The evolution rule gives, depending on the *current time*, the global store and the current state of the collective, and a control action f, a tuple of functions $\varepsilon = \langle \mu_p, \mu_w, \mu_r, \mu_u \rangle$ called the *evaluation context*. The functions μ_p and μ_w depend on the activity type α and the stores of the sender (γ_s) and receiver (γ_r) and determine the probabilities for eligible receivers to receive a message corresponding to an output action α. In the case of μ_p, the function gives a probability of a broadcast message being received successfully. In contrast μ_w deals with the unicast communication and returns a weight value. The probability of a given receiver receiving the message is obtained by normalising the weight with respect to the sum of weights of all possible receivers. The functions μ_r and μ_u depend on the activity type and the sender store. The function μ_r determines the rate

with which a given output action is performed and μ_u defines the updates on the environment (global store and collective) induced by the action.

In our example, suppose f denotes the action chosen at time t and let γ_s and γ_r denote the sender and receiver store respectively. Firstly, we define

$$\mu_p(f(\gamma_s), f(\gamma_r), \mathit{sense}^*) = 1 \qquad \text{for all stores } \gamma_s \text{ and } \gamma_r.$$

In particular, a broadcast message is received with probability 1 by all eligible receivers. There are no unicast actions in this model so the definition of μ_w is trivial. Supposing $f(\gamma_g)(\mathsf{mover}) = r_m$, $f(\gamma_g)(\mathsf{senser}) = r_s$ we can say that the rates of the actions are given as follows.

$$\mu_r(f(\gamma), \mathit{random}^*) = \mu_r(f(\gamma), \mathit{directed}^*) = r_m \qquad \text{for all local stores } \gamma$$

$$\mu_r(f(\gamma), \mathit{sense}^*) = \begin{cases} r_s & \text{for all local stores } \gamma \text{ such that } f(\gamma)(\mathsf{loc}) = f(\gamma_g)(\mathsf{tloc}) \\ 0 & \text{otherwise} \end{cases}$$

Thus, the *sense** action, in this case, is only possible from the location $(1, 1)$. The global store definitions and the composition of the collective do not change so μ_u is again trivial. This completes the description of the CARMA-C model.

3.4 System

As mentioned, a CARMA-C system is composed of a collective of components operating in an environment. For the running example we define the robot components as a pair composed of process description and a store $(\mathit{Robot}, \gamma_l)$. Finally we consider a collective of N robots denoted $(\mathit{Robot}, \gamma_l)[N]$ **in** (γ_g, ρ).

Further details on the decision process semantics are omitted here due to space constraints and can be found in [22]. We simply note that the state of the pCTMDP corresponding to the described CARMA-C model are represented by one counting variable for each considered location. The set of feasible actions in each state corresponds to the choices of the mover, senser and succp attribute values from the sets $[0, \infty)$, $[0, \infty)$ and $[0, 1]$ respectively. The value domains of the remaining attributes are trivial to resolve being defined as singleton sets.

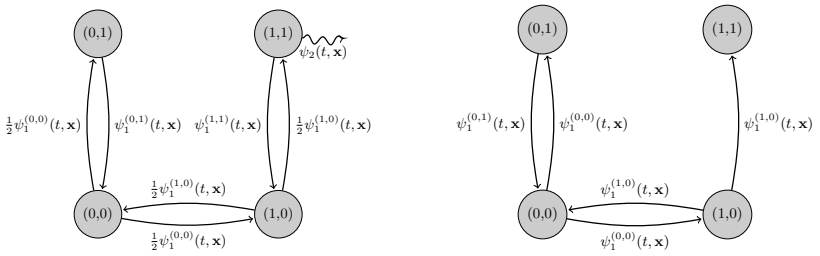
4 Case Study

4.1 Stationary target

In the previous section we gave the CARMA-C model of the robot swarm example. In this section we explore this model further. To start we describe the CTMDP model that arises if appropriately chosen semantics are applied. As discussed in Section 2 it is often useful to consider the population structure of the model.

The process state of the robots does not change throughout the evolution. Thus, the only part of each component's state that changes is the location attribute. Let us denote the state space of the pCTMDP by the counting variables

$$\mathbf{X} = (X_{01}, X_{00}, X_{10}, X_{11}, X^{11}_{01}, X^{11}_{00}, X^{11}_{10}, X^{11}_{11})$$



(a) Before broadcast – random walk (b) After broadcast – directed walk

Fig. 5: Behaviour of individuals in the swarm model with 4 locations under some deterministic policy ψ.

where X_{ij} denotes the count of robots at location (i,j) that do not know the target location while X_{ij}^{11} denotes the count of robots at location (i,j) that know that the target location is $(1,1)$.

The rates with which the actions are performed are linked to the global store variables mover and senser that are only specified through their value domains. This corresponds to the first part of the action space for the pCTMDP — at each state of the model we need to specify the particular values to be used for mover and senser. The second part of the action space corresponds to the local succp attribute. For each location we have to specify the value of succp from the interval $[0,1]$. A policy, following the definition given in Section 2, is a function

$$\psi : \mathbb{R}_{\geq 0} \times \mathbb{Z}^8_{\geq 0} \times \mathbb{R}^2_{\geq 0} \times [0,1]^8 \to [0,1]$$

assigning a probability for each of the possible combinations of attributes mover, senser and succp for each time $t \in \mathbb{R}_{\geq 0}$ and state $\mathbf{x} \in \mathbb{Z}^8_{\geq 0}$. Remember, that the choice of succp has to be made for each location giving rise to four copies of $[0,1]$ in the signature of the function. The above corresponds to the non-trivial parts of the policies ψ. To give a perfectly precise description according to the semantics the policy would also have to assign values for each of the loc and target attributes. However, as explained, the value domains for these remain singleton sets throughout the evolutions and thus the choice of policy with respect to those attributes is trivial. Denote the resulting space of probability distributions by Π. In the following we are going to consider deterministic policies such that

$$\psi : \mathbb{R}_{\geq 0} \times \mathbb{Z}^8_{\geq 0} \times \mathbb{R}^2_{\geq 0} \times [0,1]^8 \to \{0,1\}.$$

Application of a policy ψ to the pCTMDP corresponding to the model gives us behaviours of individual robots as given in Figure 5. We have denoted by $\psi_1^{(i,j)}(t,\mathbf{x})$ the rate of robots moving out of location (i,j) at time t given the population state $\mathbf{x}$ under the deterministic policy ψ. Similarly, $\psi_2(t,\mathbf{x})$ denotes the rate of sensing and broadcasting the message about the target.

Policy synthesis In this section we are going to restrict the space of policies Ψ to those that are stationary, or in other words, not dependent on time. However,

instead of having policies that map each state of the population to the same fixed value, we are going to model the situation where the movement rate of the robots decreases as the density in a given location increases. Congestion or interference is a common problem in swarm robotics that usually leads to degraded performance [17, 21, 24]. This happens especially in the cases where robots are moving towards a common target region and have to compete for available space. For this example we are considering one possible way to capture such effects on the swarm behaviour.

In order to model the congestion effects we are going to construct the policy ψ so that some maximum movement rate r_m, given by the global store attribute mover, of robots is multiplied by the exponential $e^{-a\times\frac{x}{N}}$ where x denotes the population density at the given location. In particular, the rate of movement out of location (i,j) under policy ψ becomes $\psi_1^{(i,j)}(t,\mathbf{x}) = r_m e^{-a\times\frac{x_{ij}}{N}}$, where x_{ij} is the population density at location (i,j). Such exponential degradation of the performance of individual robots in a swarm was reported, for example, in [17]. The constant a controls how fast the rate of movement decreases with the increase in number of robots in a given location. The higher values of a correspond to more severe effects of congestion. The meaning of this model would be that if the entire swarm is in the same location the congestion has the effect of approximately halving the rate of movement. [1]

In the context of the running example we consider the synthesis of the succp parameter. That is, how robust the behaviour of the robots should be for the collective to satisfy its goal. We consider the following objective: with probability greater than 0.9, 80% of the swarm reach the target location $(1,1)$ in the finite time interval $[0,10]$. We will refer to this as Obj_1.

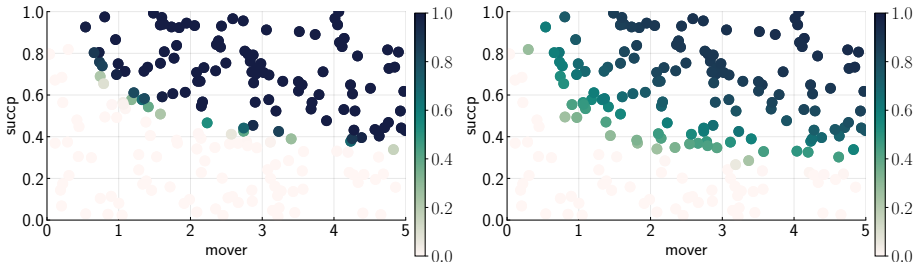
4.2 Moving Target

In this section we are going to propose and study an extension to the model considered in Section 4.1. In particular, there we assumed that the target location remains the same throughout the evolution of the system. We extend the model by considering a target whose location will change over time. To achieve that we add an extra component, named *Target* to the system. Suppose the initial state of its local store is $\gamma^{targ} = \{\mathsf{loc} \mapsto \{(1,1)\}\}$. For the movement we are going to define the following process

$$Move \stackrel{\mathsf{def}}{=} [\pi_{mt}]move^*[\circ]\langle\circ\rangle\{\mathsf{loc} \mapsto K(\mathsf{loc})\}.Move$$

where K maps locations $(1,1) \mapsto (1,2)$. The guard π_{mt} is defined to stop the target after reaching location $(1,2)$. A simple way to model that after the target has moved to a different location the robots have to look for it again is to suppose that the robots also have a process that defines the broadcast input

[1] Note that the above construction could equivalently be done directly in the definition of the rates of *random** and *directed** actions.



(a) Stationary target with Obj_1. (b) Moving target with Obj_2.

Fig. 6: Probability of success with both mover and succp varying. Constant a fixed to 0.7.

action corresponding to $move^*$.

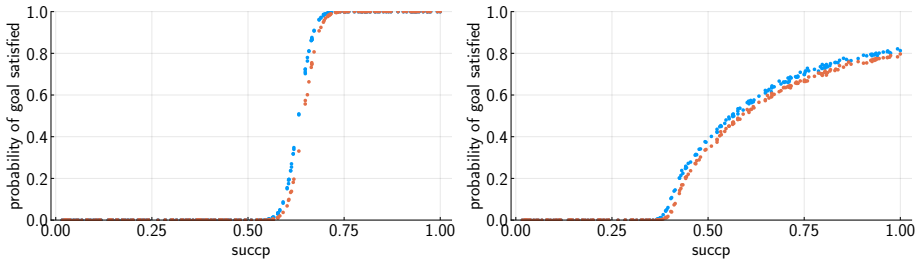
$$ListenT \stackrel{\text{def}}{=} move^*[\circ](\circ)\{\mathsf{target} \mapsto \{\emptyset\}\}.ListenT$$
$$Robot \stackrel{\text{def}}{=} (Explore \parallel Listen \parallel ListenT, \gamma)$$

In particular, when the target location changes the robots immediately know that their current set of target locations is no longer valid. To complete the extension we need to define the rate at which the target moves and the probability with which the broadcast resulting from $move^*$ is received. In our example we set the rate of $move^*$ to 0.05 and assume that all robots will be aware of when the target has left its current location. Finally, the global store update for action $move^*$ changes the value domain for the attribute tloc to correspond to the location of the target component. This ensures that after the target moves the sense actions will be available only from the new location of the target. The rest of the model remains the same. The objective, denoted Obj_2, for the new scenario is the following: with probability greater that 0.9, 80% of the swarm reach the target locations while the target is there, in the finite time interval $[0, 30]$.

4.3 Simulation results

In both of the described models we have left the exact values of succp and mover unspecified. The third parameter in the model descriptions is the congestion parameter a. This parameter would in general relate to the physical size of the considered location, size of the robots and their collision avoidance behaviour. For this simulation analysis we are going to simplify the situation by considering a range of 21 equally spaced values in the interval $[0.5, 0.9]$ and see how the results to the policy synthesis problems change with these values. Similarly, for each value of the congestion parameter we consider 10 classes of policies with each keeping the movement rate attribute mover constant in the interval $[0.0, 5.0]$.

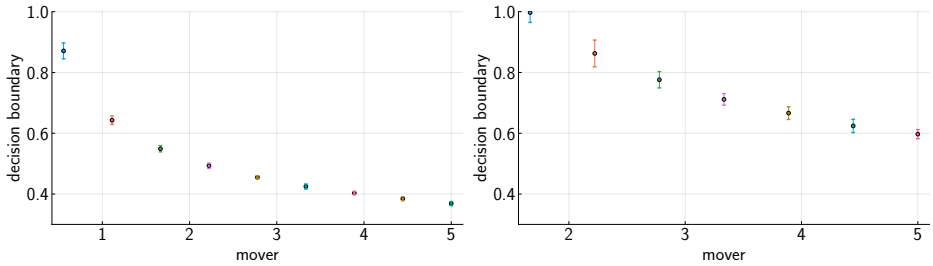
For each class of policies where the constant a and rate attribute mover are kept constant we are going to vary the values of succp. We treat the policy synthesis problem as a logistic regression problem, aiming to separate the values



(a) Satisfying Obj_1 for the stationary target example.

(b) Satisfying Obj_2 for the moving target example.

Fig. 7: Probability of success with fixed mover = 1.0 and varying succp. Blue results correspond to congestion parameter set to 0.5 while red results correspond to congestion parameter 0.9.



(a) Stationary target with Obj_1.

(b) Moving target with Obj_2.

Fig. 8: Changes in the decision boundary for logistic regression as mover changes. Points indicate the mean over the tested values of congestion constant a with error bars indicating the range of values acquired.

succp based on whether the objectives would be satisfied or not. This is done by sampling values of succp and simulating the CTMC dynamics resulting from fixing a policy of the constructed models. This is akin to works on parameter synthesis which aim to find the regions of the parameter space where a given specification is satisfied [6, 9, 5].

The approach for this is standard: consider a linear function $y = w_0 + w_1 p$ of single explanatory variable (in this case value of succp, denoted p) and a logistic function $\sigma(r) = 1/(1 + e^{-w_0 - w_1 p})$ where $\sigma(p)$ is interpreted as the probability of success given succp value p. We are going to expect the goal to be satisfied if $\sigma(p) > 0.5$. The weights for the regression model are going to be fitted based on trajectories sampled using stochastic simulation for 200 random succp values. For each of the resulting 42000 parametrisations of the model we generated 5000 trajectories using Gillespie's algorithm. Based on these trajectories we estimated the satisfaction probability for the defined objectives. Currently, the tools for CARMA do not support the non-deterministic specifications described

here. In this paper the structure of the models as well as the policies are relatively simple and for each choice of policy (or parametrisation) of the model we can readily construct a chemical reaction network model that captures the behaviour of the Carma-C model. These models were constructed and simulated with the DifferentialEquations.jl [23] package for the Julia programming language which includes methods for specifying chemical reaction networks and implements stochastic simulation algorithms for simulating the underlying CTMC.

Figure 6 shows the empirical success probabilities for a fixed value of parameter $a = 0.7$. We can see that for the simpler model with stationary target there is a quicker transition from not satisfying the objective to satisfying the objective as either of the succp or mover parameters are increased. For the example with the moving target this transition is more gradual. This observation is confirmed by Figure 7 where both the constant a and the rate of movement mover are kept constant while varying the attribute succp. In the case of the moving target example we see that setting the movement rate of the robots to 1.0 means the defined objective will not be satisfied. In both cases varying the parameter a within the rage $[0.5, 0.9]$ does not have a large effect on whether the objective can be satisfied. Finally, Figure 8 presents the results of the logistic regression being performed on the simulation data. Unsurprisingly the effect of varying the congestion constant gives a more pronounced effect on the decision boundaries for the logistic regression. Similarly, the stationary target scenario is more robust to unreliable navigation by the robots. The decision boundary for the stationary target falls below 0.5 for faster robot components. This is due to robots not moving out of the target's location when the location is known.

Finally, we make a note about the computational difficulty of treating such problems. Even for the relatively simple problems presented here, the computation time becomes large. The multi-threaded (16 threads) sampling of trajectories for the fixed policies took 5.9 hours in total for the stationary target example while the moving target simulations took about 11.4 hours.

5 Related works

There exists a large body of work on CTMDPs both from the model checking and optimisation perspectives. CTMDP models incorporate non-determinism in the model description which is usually interpreted as possible control actions that can be taken from a given state of the system. The model checking approaches seek to verify whether or not a CTMDP satisfies the requirements for a given class of policies. These commonly deal with time-bounded reachability [1, 8]. The optimisation perspective is to find a policy which maximises some utility or minimises a cost function. In both cases the core issue is scalability; statistical or simulation-based approaches offer a set of tools feasible for complex systems of collective behaviour [4, 2]. An alternative interpretation would be to consider the non-determinism as being uncertainty about parts of the system's behaviour. In the context of process algebras this idea has been considered in [13] to integrate data and uncertainty into formal quantitative models in a meaningful way.

6 Conclusion

In this paper we have presented a swarm robotics-inspired case study which presents a framework fitting together ideas from formal modelling and policy synthesis. In particular, we described a model expressed in the CARMA-C language equipped with CTMDP semantics and set up a simple policy synthesis problem where parameters can be changed or controlled. The semantics of the language presented does not discriminate against more complex cases like time-dependent or probabilistic policies. With an appropriate choice of policy space we could, for example, consider scenarios where the movement rate of the robots further degrades with time. This makes the considered framework a powerful modelling tool for stochastic control problems for collective systems. However, as seen, the statistical and simulation based approaches considered here, while in general more scalable than exact methods, are already becoming time-consuming for relatively simple problems. For the examples in this paper we may be able to decrease the number of evaluated policies for reasonable estimates but further work on approximate methods on policy synthesis for the models is of interest to reduce computational burden and allow dealing with complex policies.

References

1. Baier, C., Hermanns, H., Katoen, J., Haverkort, B.R.: Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. Theor. Comput. Sci. **345**(1), 2–26 (2005)
2. Bartocci, E., Bortolussi, L., Brázdil, T., Milios, D., Sanguinetti, G.: Policy learning in continuous-time Markov decision processes using Gaussian processes. Perform. Eval. **116**, 84–100 (2017)
3. Bernardo, M., Gorrieri, R.: Extended Markovian process algebra. In: Montanari, U., Sassone, V. (eds.) CONCUR '96: Concurrency Theory. pp. 315–330. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
4. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time markov chains. Inf. Comput. **247**, 235–253 (2016)
5. Bortolussi, L., Policriti, A., Silvetti, S.: Logic-based multi-objective design of chemical reaction networks. In: Hybrid Systems Biology - 5th International Workshop, HSB 2016, Grenoble, France, October 20-21, 2016, Proceedings. pp. 164–178 (2016)
6. Bortolussi, L., Silvetti, S.: Bayesian statistical parameter synthesis for linear temporal properties of stochastic models. In: Beyer, D., Huisman, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 396–413. Springer International Publishing (2018)
7. Brambilla, M., Brutschy, A., Dorigo, M., Birattari, M.: Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. ACM Trans. Auton. Adapt. Syst. **9**(4), 17:1–17:28 (2014)
8. Butkova, Y., Hatefi, H., Hermanns, H., Krcál, J.: Optimal continuous time Markov decisions. In: Automated Technology for Verification and Analysis, 2015, Proceedings. pp. 166–182 (2015)
9. Ceska, M., Dannenberg, F., Paoletti, N., Kwiatkowska, M., Brim, L.: Precise parameter synthesis for stochastic biochemical systems. Acta Inf. **54**(6), 589–623 (2017)

10. De Nicola, R., Latella, D., Loreti, M., Massink, M.: A uniform definition of stochastic process calculi. ACM Comput. Surv. **46**(1), 5:1–5:35 (2013)
11. Galpin, V.: Modelling ambulance deployment with Carma. In: Coordination Models and Languages - 18th IFIP WG 6.1 International Conference, COORDINATION 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings. pp. 121–137 (2016)
12. Galpin, V., Zon, N., Wilsdorf, P., Gilmore, S.: Mesoscopic modelling of pedestrian movement using CARMA and its tools. ACM Transactions on Modeling and Computer Simulation **28**(2) (2018)
13. Georgoulas, A., Hillston, J., Milios, D., Sanguinetti, G.: Probabilistic programming process algebra. In: Quantitative Evaluation of Systems - 11th International Conference, QEST 2014. pp. 249–264 (2014)
14. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press, New York, NY, USA (1996)
15. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: probabilistic symbolic model checker. In: Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS. Proceedings. pp. 200–204. Springer (2002)
16. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Runtime Verification - First International Conference, RV. Proceedings. pp. 122–135. Springer (2010)
17. Lerman, K., Galstyan, A.: Mathematical model of foraging in a group of robots: Effect of interference. Auton. Robots **13**(2), 127–141 (2002)
18. Loreti, M., Hillston, J.: Modelling and analysis of collective adaptive systems with CARMA and its tools. In: Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems, Advanced Lectures. pp. 83–119 (2016)
19. Luckcuck, M., Farrell, M., Dennis, L.A., Dixon, C., Fisher, M.: Formal specification and verification of autonomous robotic systems: A survey. ACM Comput. Surv. **52**(5) (2019)
20. Lv, H., Hillston, J., Piho, P., Wang, H.: An attribute-based availability model for large scale IaaS clouds with CARMA. IEEE Trans. Parallel Distrib. Syst. **31**(3), 733–748 (2020)
21. Marcolino, L.S., dos Passos, Y.T., de Souza, Á.A.F., dos Santos Rodrigues, A., Chaimowicz, L.: Avoiding target congestion on the navigation of robotic swarms. Auton. Robots **41**(6), 1297–1320 (2017)
22. Piho, P., Hillston, J.: Policy synthesis for collective dynamics. In: 15th International Conference on Quantitative Evaluation of SysTems (QEST 2018). Springer (6 2018)
23. Rackauckas, C., Nie, Q.: DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. Journal of Open Research Software **5** (05 2017)
24. Schroeder, A., Trease, B., Arsie, A.: Balancing robot swarm cost and interference effects by varying robot quantity and size. Swarm Intelligence **13**(1), 1–19 (2019)
25. Zon, N., Gilmore, S.: Data-driven modelling and simulation of urban transportation systems using CARMA. In: Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems - 8th International Symposium, ISoLA 2018, Cyprus, Nov5-9, 2018, Proceedings, Part III. pp. 274–287 (2018)